📘 React `useState`, Props, and Parent-Child Hierarchy

(Based on the given code structure)

🧠 Conceptual Understanding

✅ What is `useState`?

- `useState` is a React Hook that lets you add **state** to functional components.
- Syntax:

```
const [stateVariable, setStateFunction] = useState(initialValue);
```

✅ What are `props`?

- `props` (short for *properties*) allow you to pass data from a **parent component** to a **child component**.
- They are **read-only** in the child, but the parent can pass functions too, which allow child to **modify parent's state**.

✅ Parent-Child Hierarchy in this example

- `App.jsx` is the **parent**.
- `Searchbox.jsx` is the **child**.
- Two child components are created and rendered from the same component definition but with different identifiers (`c_number = "first"` / `"second"`).

⚙️ Code Breakdown

📂 App.jsx (Parent Component)

```
import { useState } from 'react'
import './App.css'
import Searchbox from './components/searchbox'

function App() {
  const [sentence, changeSentence] = useState('');

  return (
    <>
      <Searchbox
        c_number = "first"
        sentence={sentence}
        changeSentence={changeSentence}
      ></Searchbox>
      <Searchbox
        c_number="second"
        sentence={sentence}
        changeSentence={changeSentence}
      ></Searchbox>
    </>
  );
}

export default App
```

🔍 Key Points:

- `useState` is initialized with an empty string.
- The state value `sentence` and the function `changeSentence` are passed to both `Searchbox` components via props.

- Both children are **sharing** the same state — any change in one will reflect in the other.

📂 Searchbox.jsx (Child Component)

```
import React from 'react'
import './searchbox.css'

const Searchbox = (props) => {
  return (
    <div className="Containerdiv">
      <input
        type="text"
        onChange={(e) => {
          props.changeSentence(e.target.value);
        }}
      />
      <p>You have typed: {props.sentence}</p>
    </div>
  );
}

export default Searchbox
```

🔍 Key Points:

- `props.changeSentence` is called on every `input` change — this updates the parent state.
- Since **both children share the same props**, the typed text reflects in both `Searchbox` components at the same time.
- This showcases **unidirectional data flow**:

  `Child → calls function → updates Parent → sends updated state → reflects in all children`.

🎯 Takeaways from the Video

Concept	Explained
`useState`	Used in parent (`App.jsx`) to manage a shared state.
`props`	Sent from parent to children (`sentence` , `changeSentence`).
Function as Prop	`changeSentence` passed down to be triggered by the child.
Reusability	Same `Searchbox` component reused twice with different `c_number` .
State Sharing	Both components reflect the same state — typing in one updates both.
Parent-Child Communication	Child sends data to parent using function, parent passes updated data back to children via props.

💡 Visual Summary

```
    App.jsx (Parent)
      |
  ---------------------
  |                   |
Searchbox          Searchbox
 (child 1)          (child 2)
   \                  /
    --------props---------
           |
    shared state update
```